Hacking Street Fighter:
CPS-2 Encryption in Radare2

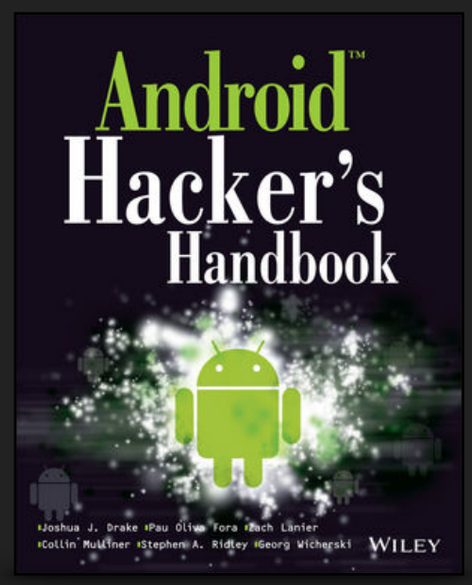
Pau Oliva Fora - @pof

CON 2016

$ whoami

- Pau Oliva Fora, aka @pof
- Security Consultant with IOActive
- R+D Engineering background:
  - Smartphone Research since 2004
  - Android Research since 2008

- Speaker at a variety of security conferences, including DefCon and RSA in USA, Android Security Symposium in Austria and OWASP, NoConName, RootedCon and LaCon in Spain
- Co-author of Android Hacker's Handbook
- Casual Super Street Fighter 2X Player :)
- Developer of FightCade



Presentation Agenda

- CPS2: Capcom Play System 2
    - What is it?, history, security overview...

- Super Street Fighter 2X
    - Debugging, patching...

- CPS2 and Radare2 <3
    - CPS2 crypto support, demos...

CPS2: Capcom Play System 2

CPS2: A + B board

CPS2: Specs

- Primary CPU: Motorola 68000 @ 16 MHz
- Sound CPU: Z80 @ 8 MHz
- Display: 384x224 @ 59.6294 Hz

CPS2: History

- CPS-1 games were easy to copy & bootlegs (unauthorised game copies) appeared
  - (02/1991) Street Fighter II: The World Warrior
  - (03/1992) Street Fighter II': Champion Edition
  - (12/1992) Street Fighter II' Turbo: Hyper Fighting

- CPS-2 == CPS-1 with a faster processor and encrypted game ROMs
  - (09/1993) Super Street Fighter II: The New Challengers
  - (02/1994) Super Street Fighter II Turbo
  - (12/2003) Hyper Street Fighter II: The Anniversary Edition

CPS2: Suicide Battery (1)

- The CPS-2 ‘B’ boards hold a battery-backed memory (SRAM) containing decryption keys needed for the games to run

3.6V Lithium battery
Size: 1/2 AA
(Elfa part #69-282-12)

- When the battery dies, the game will no longer work --> blue screen

CPS2: Suicide Battery (2)

CPS2: Encryption (1)

- In January 2001, the **CPS-2 Shock group** (Razoola and CrashTest) with Charles MacDonald, obtained unencrypted program data by hacking into the hardware

- They distributed **XOR difference tables** (8GiB) to produce unencrypted data from the original ROM images --> Emulation possible

CPS2: Encryption (2)

- In January 2007, the encryption was fully reverse-engineered by Andreas Naive and Nicola Salmoria (Mame author).

- The encryption only affects opcodes, not data.

- The encryption consists of two 4-round Feistel networks with a 64-bit key and involves both the 16-bit opcode and the low 16 bits of the address.

- The algorithm was implemented for all CPS-2 games in MAME.

CPS2: Memory Map

- `0x000000 - 0x3FFFFF` Main Program
- `0x400000 - 0x40000A` Encryption (the battery memory)
- `0x618000 - 0x619FFF` Shared RAM for the Z80 (tells what sfx or music to play)
- `0x660000 - 0x663FFF` Network Memory
- `0x900000 -` Start of Graphic memory (can change with each game)

- Super Turbo:
- `0x900000 - 0x903FFF` Palette
- `0x904000 - 0x907FFF` 16x16
- `0x908000 - 0x90BFFF` 32x32
- `0x90C000 - 0x90FFFF` 8x8
- `0x910000 - 0x913FFF` 16x16 mainly hud and character names on select screen

- `0xFF0000 - 0xFFFFFF` Main Memory

CPS2: Revive Dead B-Boards (1)

- Decrypt all encrypted data so that you end up with a fully decrypted ROM image.

- Patch all read and writes to the 0x400000-0x40000A memory region to 0xFFFFF0-0xFFFFFA (bottom of the normal WORK RAM)

- Patch all routines not to clear this region during any memory clearing activities

- Patch any part of the game code that uses this region of WORK RAM to use a different region.

CPS2: Revive Dead B-Boards (2)

- Reprogram the EPROMs with the decrypted ROM images

- Desolder/Remove the Battery (bottom right corner of the board)

- Short the 2 leads of the electrolytic capacitor next to where the + terminal was together for several seconds.

- Boot up the game, cross fingers :)

CPS2: Revive Dead B-Boards (3)

- Phoenix Edition "Decrypted" ROMs
  - Created by Razoola
  - Include some patches like region change & jukebox

- Avalaunch "Decrypted" ROMs
  - Created by Team Avalaunch (L_Oliveira, MottZilla and idc)
  - No extra features

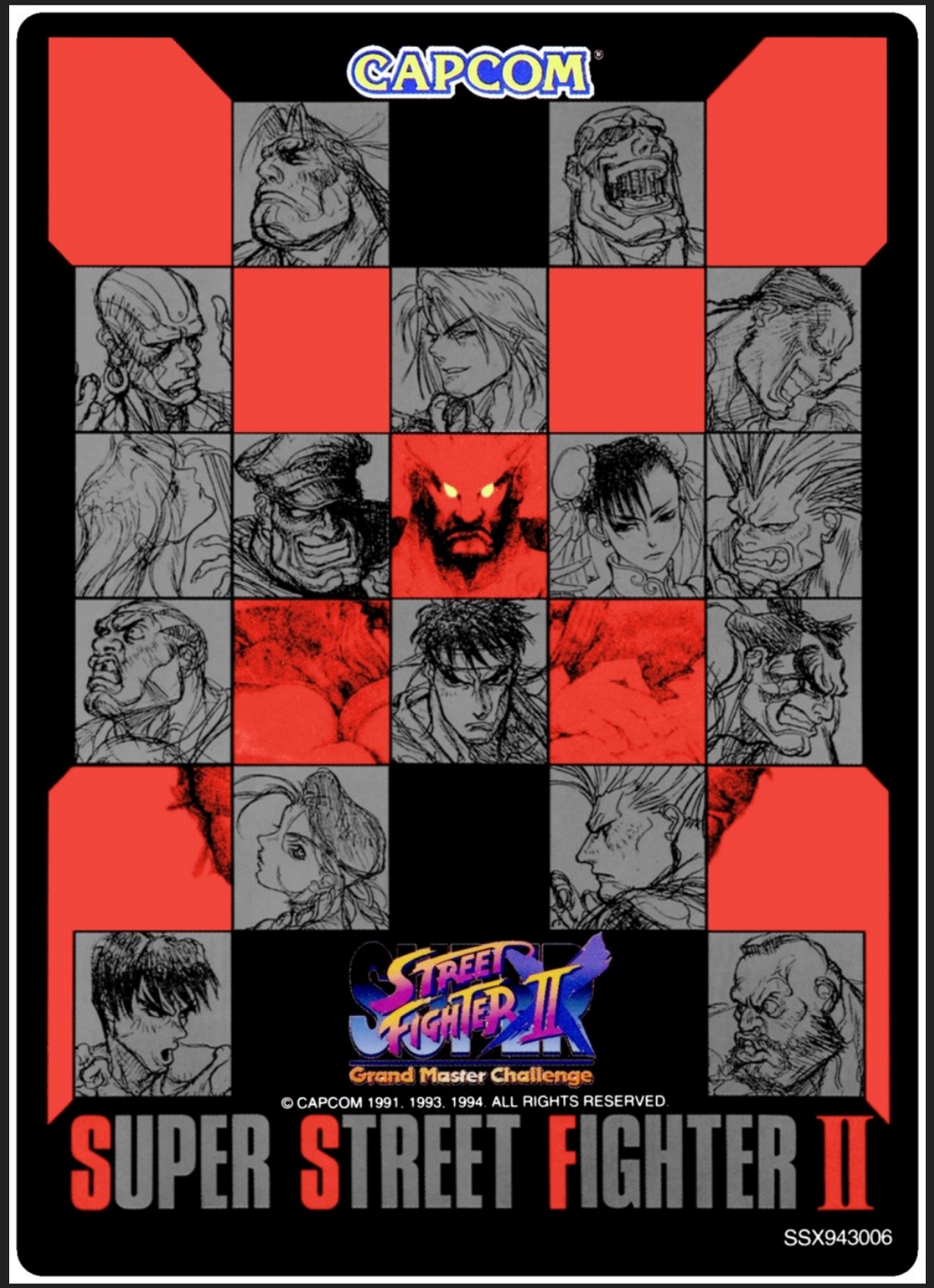
CPS2: Revive Dead B-Boards

- In April 2016, Artemio Urbina, Ian Court and Eduardo Cruz successfully reverse engineered the Capcom's CPS2 security programming, making possible a clean desuicide and restoration of any dead games without hardware modifications.

CPS2: Security Timeline

- CPS2 Released
- XOR Diff Tables (+8 years)
- Encryption keys obtained (+14 years)
- Security programming RE (+23 years)

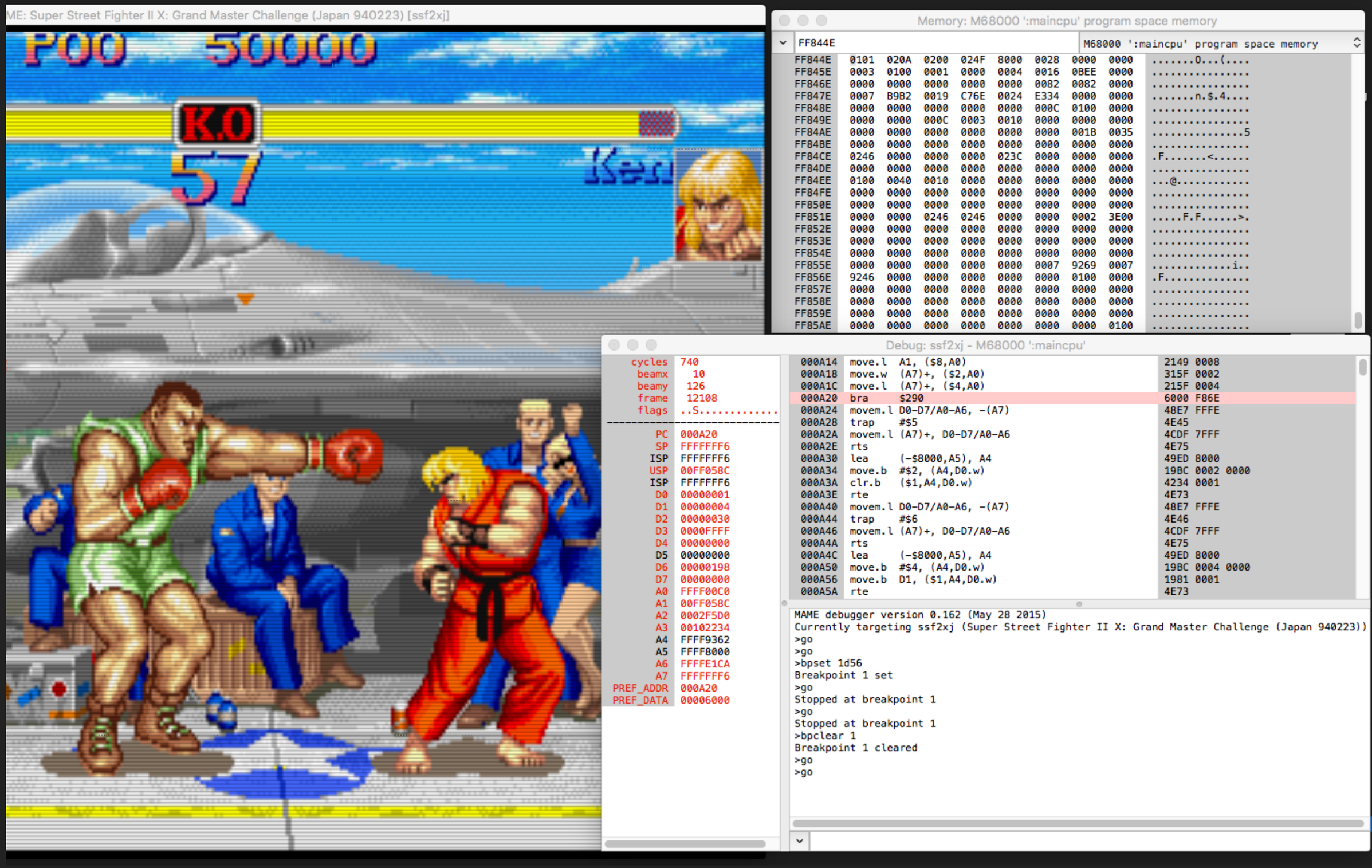
Super Street Fighter 2X



SSF2X: Debugging

- mame -debug ssf2xj
- Ctrl+M (Cmd+D on Mac) to open memory window
- Adress 0xFF844E
- Offset for P2 base is 0x400

SSF2X: Debugging



SSF2X: Lua Scripting (1)

- mame-rr –lua
- memory.readbyte(), memory.readword(),
- memory.writebyte(), memory.writeword()
- gui.text(), emu.frameadvance()

SSF2X: Lua Scripting (2)

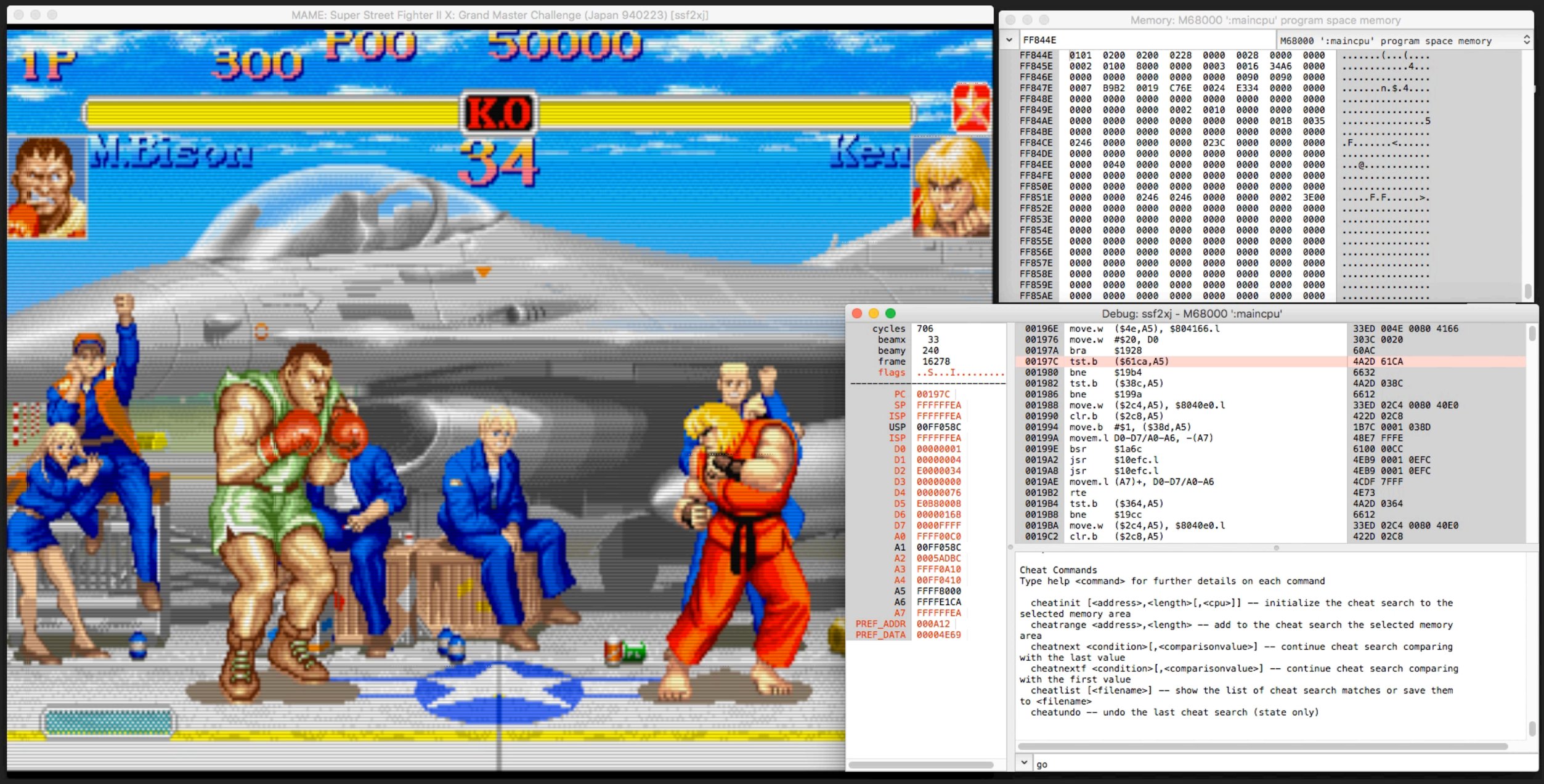
```lua
local function draw_messages()

	if memory.readword(0xFF847F) == 0 then --if not in match
		gui.text(0,0,"")
		return
	end

	if not player_names then
		gui.text(0,0,"")
		return
	end

	local p1_info = memory.readbyte(0xFF844E+0x0b)
	local p2_info = memory.readbyte(0xFF844E+0x400+0x0b)

	gui.text(34,45,p1_info)
	gui.text(339,45,p2_info)

	if (p1_info==0xc or p1_info==0x26 or p1_info==0x4a or p1_info==0x24) then
		gui.text(34,55,"BLOCK HIGH")
	end
	if (p1_info==0x28) then
		gui.text(34,55,"BLOCK LOW")
	end

	if (p2_info==0xc or p2_info==0x26 or p2_info==0x4a or p2_info==0x24) then
		gui.text(339,55,"BLOCK HIGH")
	end
	if (p2_info==0x28) then
		gui.text(310,55,"BLOCK LOW")
	end

-- 0C = Can only be blocked high (Aerial move/overhead)
-- 26 = Can only be blocked high, Full KD
-- 28 = Can only be blocked low, Forces Standing Fierce/Rh hitstun/pushback, Full KD against aerial opponents only
-- 4a = Juggle able [3-hit limit], Can only be blocked high (Ryu/Dic's j.strong)

	return
end
```

SSF2X: Cheats

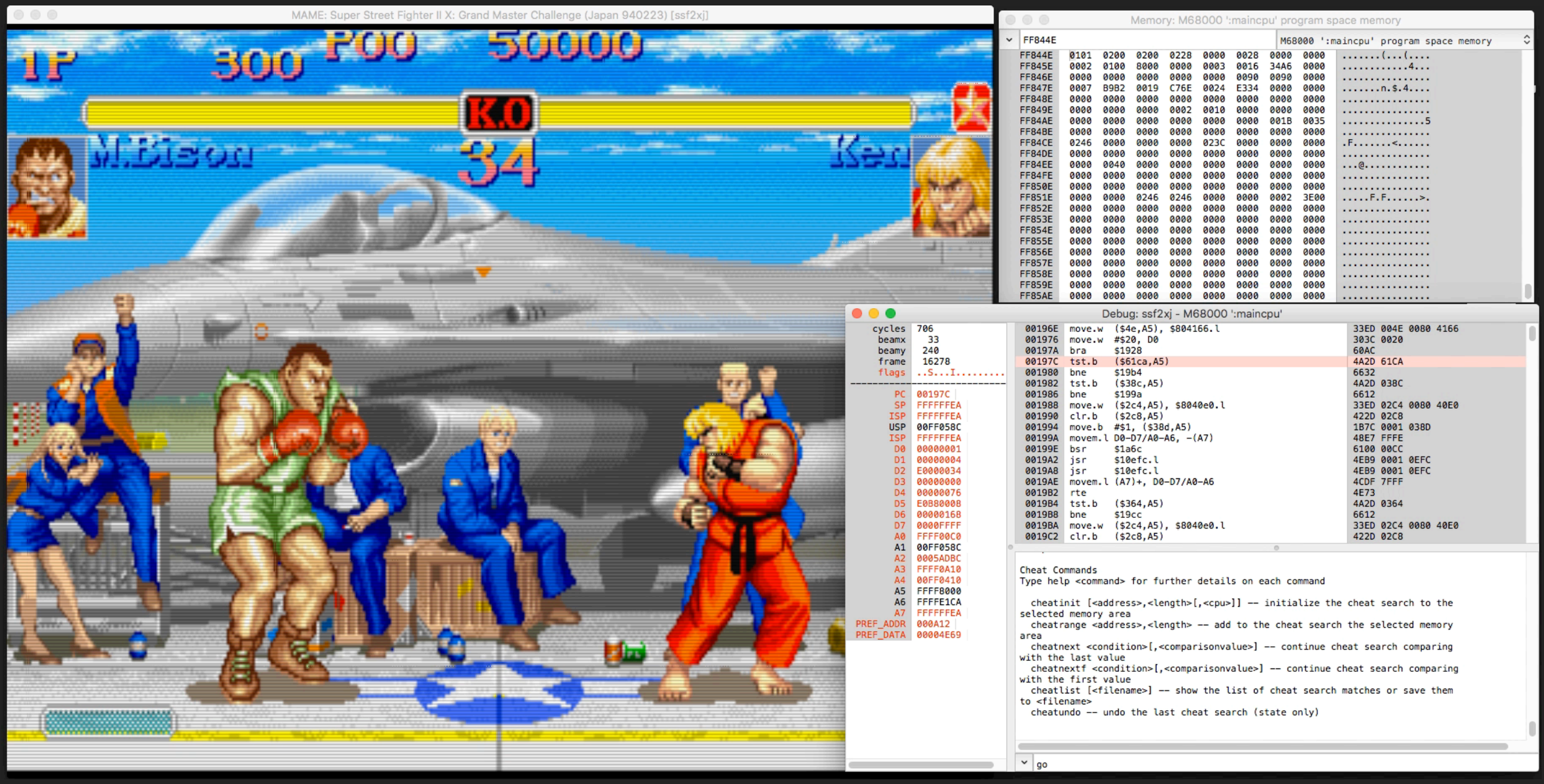
- **RAM cheats** usually change the data the game has in RAM (ie: change the value in a fixed memory address)

- **ROM cheats** patch the game's program code to force the game engine take a different path

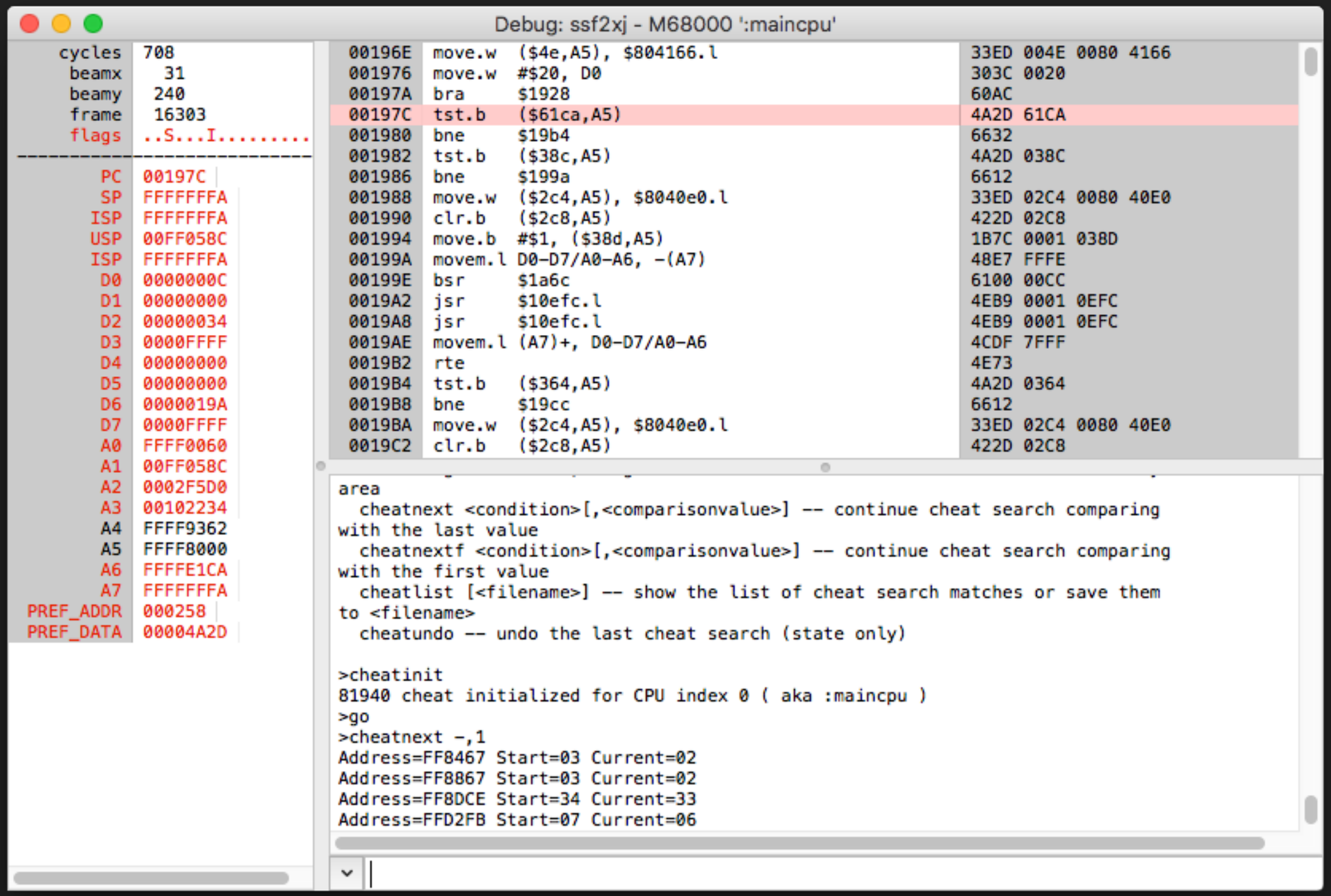
SSF2X: MAME Debugger Demo (1)



SSF2X: MAME Debugger Demo (1)



SSF2X: MAME Debugger Demo (2)



search for all bytes that have decreased by one since we did the *cheatinit* command

SSF2X: MAME Cheats (1)

```
<cheat desc="Infinite Time">
    <script state="run">
        <action>maincpu.pb@FF8DCE=99</action>
    </script>
</cheat>
```

1. maincpu: This is the tag of the CPU whose memory you want to poke, maincpu is in 99% of cases the tag you will need

SSF2X: MAME Cheats (2)

```
<cheat desc="Infinite Time">
    <script state="run">
        <action>maincpu.pb@FF8DCE=99</action>
    </script>
</cheat>
```

2. p : memory space that needs to be poked, there are 7 possibilities:
    p = program write (most RAM cheats need this)
    m = region write (most ROM cheats use this)
    r = RAM write
    o = Opcode Write (often used for encrypted memory)
    d = data write
    i = i/o write
    3 = SPACE3 write

SSF2X: MAME Cheats (3)

```
<cheat desc="Infinite Time">
    <script state="run">
        <action>maincpu.pb@FF8DCE=99</action>
    </script>
</cheat>
```

3. b : memory size of what's being poked, there are 4 possibilities:
    b (byte)
    w (word=2 bytes)
    d (doubleword=4 bytes)
    q (quadword=8 bytes)

SSF2X: MAME Cheats (4)

```
<cheat desc="Infinite Energy P1">
    <script state="run">
        <action>maincpu.pw@FF8478=90</action>
    </script>
</cheat>
```

- More examples: https://github.com/poliva/ssf2xj

SSF2X: Debugger Watchpoints (1)

Memory: M68000 ':maincpu' program space memory

ff844e — M68000 ':maincpu' program space memory

FF844E	0101	0200	0200	0228	0000	0028	0000	0000	.....(...(....
FF845E	0002	0100	0000	0000	0004	0011	2202	0000	..........."...
FF846E	0000	0000	0000	0000	0000	0090	0090	0000	....
FF847E	0007	B912	0019	8832	0024	E1F4	0000	0000	.......2.$......
FF848E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF849E	0000	0000	0000	0002	0010	0000	0000	0000	....
FF84AE	0000	0000	0000	0000	0000	0000	001D	0035	5
FF84BE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84CE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84DE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84EE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84FE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF850E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF851E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF852E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF853E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF854E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF855E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF856E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF857E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF858E	0000	0000	0000	0000	0000	0000	0000	0000	....

Debug: ssf2xj - M68000 ':maincpu'

0C6BD6	beq	$c6bca	67F2
0C6BD8	lea	(-$5e72,A5), A1	43ED A18E
0C6BDC	bra	$c6e6a	6000 028C
0C6BE0	move.b	($154d,A5), ($1557,A5)	1B6D 154D 1557
0C6BE6	move.w	($1582,A5), ($1592,A5)	3B6D 1582 1592
0C6BEC	move.b	($154d,A5), D0	102D 154D
0C6BF0	beq	$c6c06	6714
0C6BF2	movea.w	($1582,A5), A4	386D 1582
0C6BF6	movea.w	(A4)+, A1	325C
0C6BF8	bsr	$c6e6a	6100 0270
0C6BFC	subq.b	#1, ($154d,A5)	532D 154D
0C6C00	bne	$c6bf6	66F4
0C6C02	move.w	A4, ($1582,A5)	3B4C 1582
0C6C06	rts		4E75
0C6C08	move.b	($154e,A5), ($1558,A5)	1B6D 154E 1558
0C6C0E	move.w	($1584,A5), ($1594,A5)	3B6D 1584 1594
0C6C14	move.b	($154e,A5), D0	102D 154E
0C6C18	beq	$c6c2e	6714
0C6C1A	movea.w	($1584,A5), A4	386D 1584
0C6C1E	movea.w	(A4)+, A1	325C

```
Watchpoint Commands
Type help <command> for further details on each command

  wp[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets program
space watchpoint
  wpd[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets data
space watchpoint
  wpi[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets I/O space
watchpoint
  wpclear [<wpnum>] -- clears a given watchpoint or all if no <wpnum> specified
  wpdisable [<wpnum>] -- disables a given watchpoint or all if no <wpnum>
specified
  wpenable [<wpnum>] -- enables a given watchpoint or all if no <wpnum>
specified
  wplist -- lists all the watchpoints
  hotspot [<cpu>,[<depth>[,<hits>]]] -- attempt to find hotspots
```

```
wpset 0xFF8878,1,w,1,{printf "P2 Write @ %X=%X with PC=%X", wpaddr, pw@FF8878, PC; go}
```

SSF2X: Debugger Watchpoints (1)

Memory: M68000 ':maincpu' program space memory

ff844e — M68000 ':maincpu' program space memory

FF844E	0101	0200	0200	0228	0000	0028	0000	0000	.....(...(....
FF845E	0002	0100	0000	0000	0004	0011	2202	0000	..........."...
FF846E	0000	0000	0000	0000	0000	0090	0090	0000	....
FF847E	0007	B912	0019	8832	0024	E1F4	0000	0000	.......2.$......
FF848E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF849E	0000	0000	0000	0002	0010	0000	0000	0000	....
FF84AE	0000	0000	0000	0000	0000	0000	001D	0035	5
FF84BE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84CE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84DE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84EE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF84FE	0000	0000	0000	0000	0000	0000	0000	0000	....
FF850E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF851E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF852E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF853E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF854E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF855E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF856E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF857E	0000	0000	0000	0000	0000	0000	0000	0000	....
FF858E	0000	0000	0000	0000	0000	0000	0000	0000	....

Debug: ssf2xj - M68000 ':maincpu'

0C6BD6	beq	$c6bca	67F2
0C6BD8	lea	(-$5e72,A5), A1	43ED A18E
0C6BDC	bra	$c6e6a	6000 028C
0C6BE0	move.b	($154d,A5), ($1557,A5)	1B6D 154D 1557
0C6BE6	move.w	($1582,A5), ($1592,A5)	3B6D 1582 1592
0C6BEC	move.b	($154d,A5), D0	102D 154D
0C6BF0	beq	$c6c06	6714
0C6BF2	movea.w	($1582,A5), A4	386D 1582
0C6BF6	movea.w	(A4)+, A1	325C
0C6BF8	bsr	$c6e6a	6100 0270
0C6BFC	subq.b	#1, ($154d,A5)	532D 154D
0C6C00	bne	$c6bf6	66F4
0C6C02	move.w	A4, ($1582,A5)	3B4C 1582
0C6C06	rts		4E75
0C6C08	move.b	($154e,A5), ($1558,A5)	1B6D 154E 1558
0C6C0E	move.w	($1584,A5), ($1594,A5)	3B6D 1584 1594
0C6C14	move.b	($154e,A5), D0	102D 154E
0C6C18	beq	$c6c2e	6714
0C6C1A	movea.w	($1584,A5), A4	386D 1584
0C6C1E	movea.w	(A4)+, A1	325C

```
Watchpoint Commands
Type help <command> for further details on each command

  wp[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets program
space watchpoint
  wpd[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets data
space watchpoint
  wpi[set] <address>,<length>,<type>[,<condition>[,<action>]] -- sets I/O space
watchpoint
  wpclear [<wpnum>] -- clears a given watchpoint or all if no <wpnum> specified
  wpdisable [<wpnum>] -- disables a given watchpoint or all if no <wpnum>
specified
  wpenable [<wpnum>] -- enables a given watchpoint or all if no <wpnum>
specified
  wplist -- lists all the watchpoints
  hotspot [<cpu>,[<depth>[,<hits>]]] -- attempt to find hotspots
```

```
wpset 0xFF8878,1,w,1,{printf "P2 Write @ %X=%X with PC=%X", wpaddr, pw@FF8878, PC; go}
```

SSF2X: Debugger Watchpoints (2)

Debug: ssf2xj - M68000 ':maincpu'

```
    cycles  750
     beamx    3
     beamy  241
     frame   9440
     flags  ..S...I....X....
-----------------------------
        PC  001B7C
        SP  FFFFFFBA
       ISP  FFFFFFBA
       USP  00FF058C
       ISP  FFFFFFBA
        D0  FFFF003F
        D1  0000807D
        D2  00000000
        D3  00000000
        D4  00000000
        D5  00000000
        D6  0000019F
        D7  00000000
        A0  FFFF0200
        A1  00FF058C
        A2  0002F5D0
        A3  00102234
        A4  FFFF9362
        A5  FFFF8000
        A6  FFFFE1CA
        A7  FFFFFFBA
 PREF_ADDR  001B7C
 PREF_DATA  0000102D
```

```
001B64  move.w  ($4e,A5), $804166.l      33ED 004E 0080 4166
001B6C  move.w  ($54,A5), $400004.l      33ED 0054 0040 0004
001B74  move.w  ($56,A5), $400006.l      33ED 0056 0040 0006
001B7C  move.b  ($2e7,A5), D0            102D 02E7
001B80  move.b  D0, D1                   1200
001B82  lsl.b   #2, D1                   E509
001B84  andi.w  #$30, D1                 0241 0030
001B88  andi.w  #$3, D0                  0240 0003
001B8C  or.w    D1, D0                   8041
001B8E  or.w    ($34c,A5), D0            806D 034C
001B92  or.w    ($34e,A5), D0            806D 034E
001B96  tst.b   ($354,A5)                4A2D 0354
001B9A  bne     $1ba2                    6606
001B9C  move.w  D0, $804040.l            33C0 0080 4040
001BA2  move.l  ($7e,A5), D0             202D 007E
001BA6  lsr.l   #8, D0                   E088
001BA8  move.l  D0, ($7e,A5)             2B40 007E
001BAC  move.w  $804030.l, ($350,A5)     3B79 0080 4030 0350
001BB4  move.w  $804020.l, D0            3039 0080 4020
001BBA  not.w   D0                       4640
```

```
  wplist -- lists all the watchpoints
  hotspot [<cpu>,[<depth>[,<hits>]]] -- attempt to find hotspots

>wpset 0xFF8878,1,w,1,{printf "P2 Write @ %X=%X with PC=%X", wpaddr, pw@FF8878, PC; go}
Watchpoint 5 set
P2 Write @ FF8878=90 with PC=BE64A
P2 Write @ FF8878=8A with PC=7AD0C
P2 Write @ FF8878=8A with PC=BE64A
P2 Write @ FF8878=81 with PC=7AD0C
P2 Write @ FF8878=81 with PC=BE64A
P2 Write @ FF8878=75 with PC=7AD0C
P2 Write @ FF8878=75 with PC=BE64A
P2 Write @ FF8878=5F with PC=7AD0C
P2 Write @ FF8878=5F with PC=7AB3C
P2 Write @ FF8878=3F with PC=BE64A
P2 Write @ FF8878=3B with PC=7AD0C
P2 Write @ FF8878=3B with PC=BE64A
```

wpset <address>,<length>,<type>[,<condition>[,<action>]]

wpset 0xFF8878,1,w,1,{printf "P2 Write @ %X=%X with PC=%X", wpaddr, pw@FF8878, PC; go}

SSF2X: Patching m68k for dummies (1)

- **NOP** = **0x4e71**
- **BEQ** = **0x67**XXYYYYYZZZZ where XXYYYYYZZZZ indicates how far we will jump forward if the previous comparison instruction (usually a **TST**) was found to be equal.
- **BNE** = **0x66**XXYYYYYZZZZ where XXYYYYYZZZZ indicates how far we will jump forward if the previous comparison instruction (usually a **TST**) was not equal.

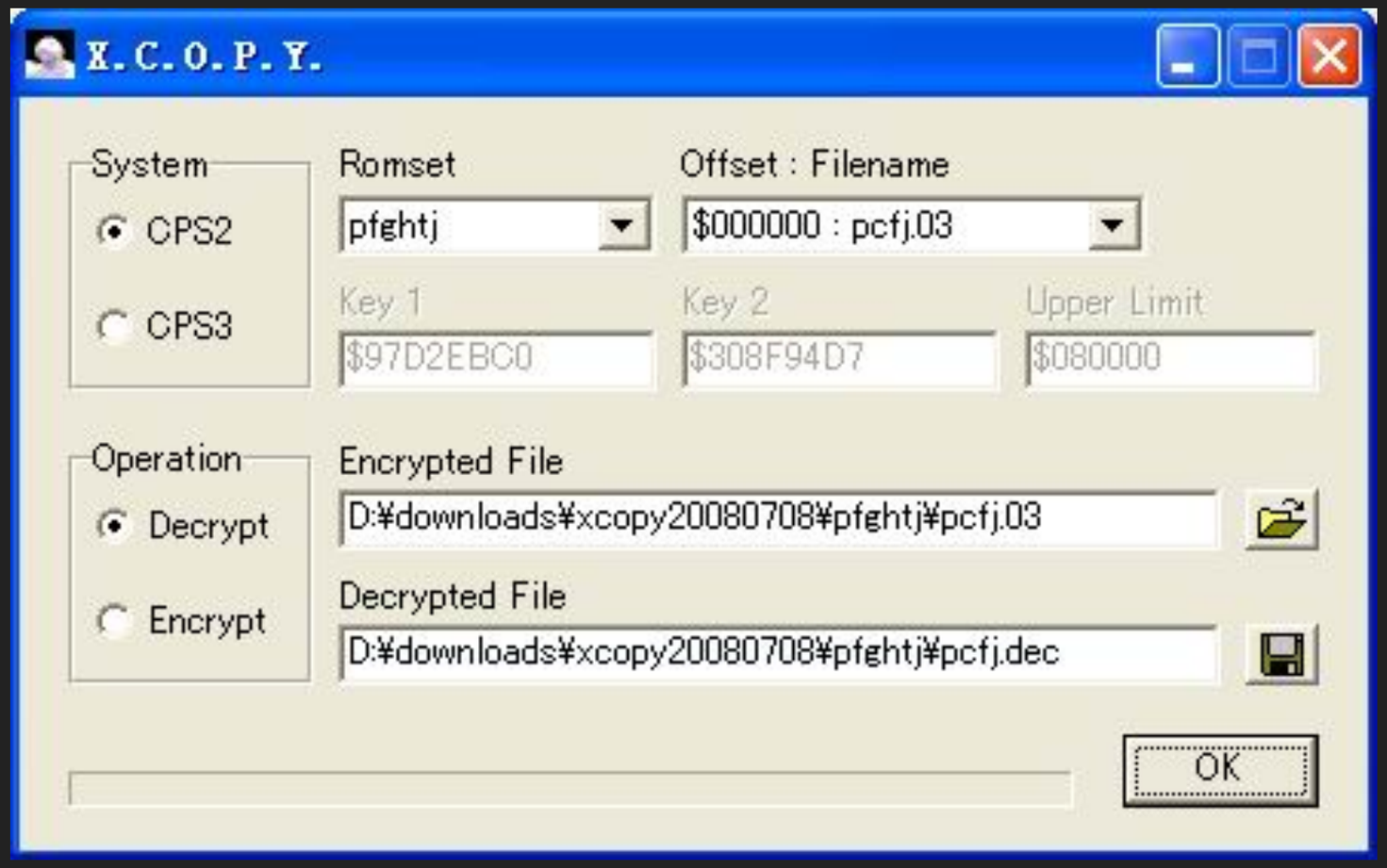
- So if we need to invert the logic we can change the **BEQ** for **BNE** by swapping a **67** for a **66** on the first byte of the opcode.
- If we want to always force a certain code path we can just **NOP** the branch instruction

SSF2X: Patching m68k for dummies (2)

```
[0x00068f38]> pD
            0x00068f38    0998          bclr d4,(a0)+
            0x00068f3a    4a2d02e6      tst.b 0x2e6(a5)
         ,=< 0x00068f3e    661a          bne.b 0x68f5a
         |   0x00068f40    4a2d0349      tst.b 0x349(a5)
        ,==< 0x00068f44    6614          bne.b 0x68f5a
        ||   0x00068f46    41ed07dc      lea 0x7dc(a5),a0
        ||   0x00068f4a    4a2e0021      tst.b 0x21(a6)
       ,===< 0x00068f4e    6704          beq.b 0x68f54
       |||   0x00068f50    41ed0bdc      lea 0xbdc(a5),a0
       `---> 0x00068f54    4a280000      tst.b 0(a0)
      ,====< 0x00068f58    671a          beq.b 0x68f74
      |  ``-> 0x00068f5a    102c0291      move.b 0x291(a4),d0
     ,=====< 0x00068f5e    6714          beq.b 0x68f74
     ||      0x00068f60    542e0002      addq.b #0x2,0x2(a6)
     ||      0x00068f64    1d7c0078001e  move.b #0x78,0x1e(a6)
     ||      0x00068f6a    2d6e00800006  move.l 0x80(a6),0x6(a6)
     ||      0x00068f70    60000038      bra.w 0x68faa
     ``----> 0x00068f74    4e75          rts
             0x00068f76    532e001e      subq.b #0x1,0x1e(a6)
             0x00068f7a    6728          beq.b 0x68fa4
             0x00068f7c    102c0291      move.b 0x291(a4),d0
    ,======< 0x00068f80    6710          beq.b 0x68f92
    |        0x00068f82    1d7c0078001e  move.b #0x78,0x1e(a6)
    |        0x00068f88    2d6e00800006  move.l 0x80(a6),0x6(a6)
    |        0x00068f8e    6100001a      bsr.w 0x68faa
    `------> 0x00068f92    202e0084      move.l 0x84(a6),d0
             0x00068f96    90ae0006      sub.l 0x6(a6),d0
```

CPS2 Encrypt / Decrypt state of the art

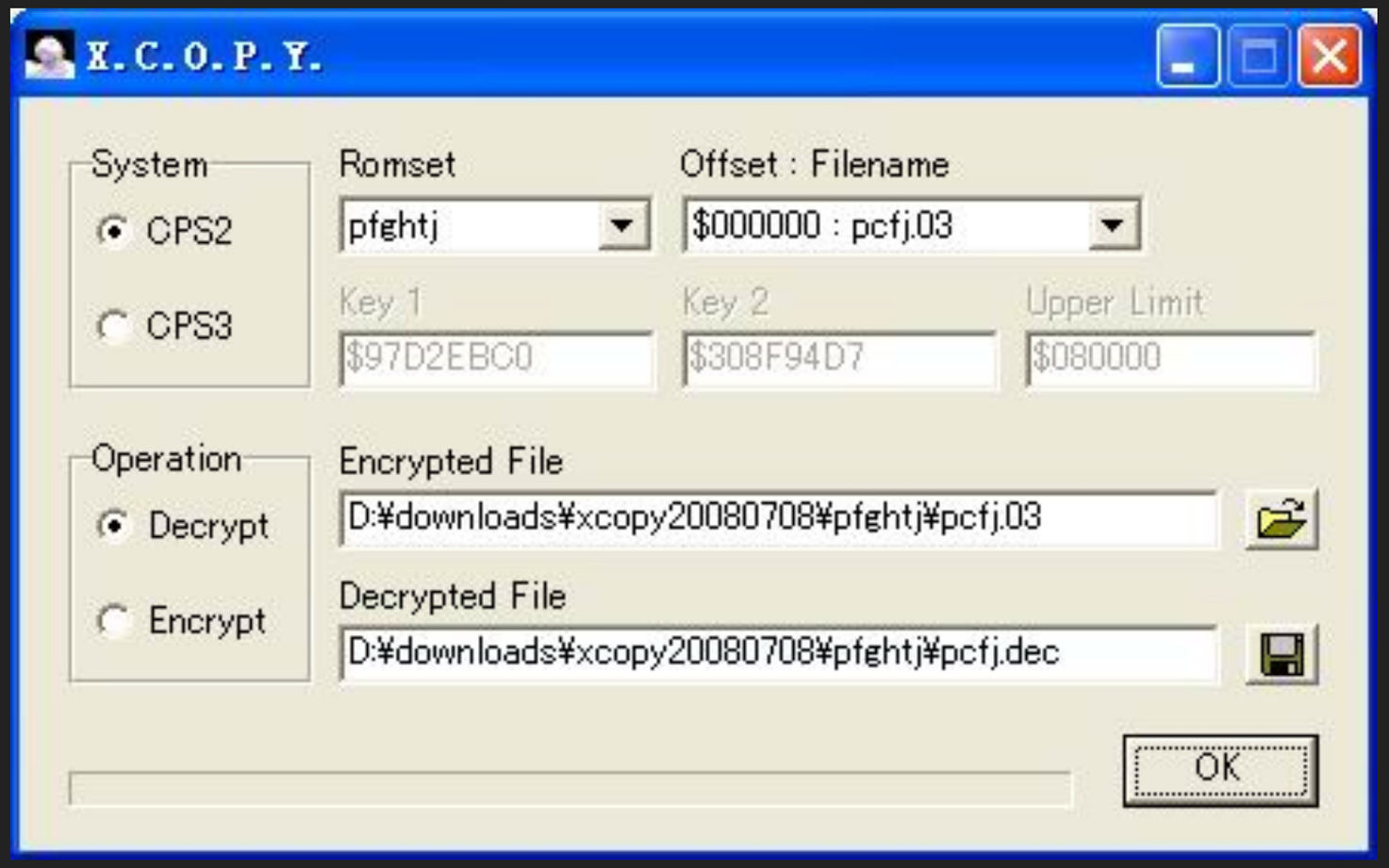
- To my knowledge, the only tool that allows to decrypt & encrypt CPS2 ROMs for rom hacking purposes is X.C.O.P.Y.
- Released by 'yumeji' in 2007, but website no longer available (geocities.jp).
- Need to dig on shady forums to find a working copy



CPS2 Encrypt / Decrypt state of the art

- To my knowledge, the only tool that allows to decrypt & encrypt CPS2 ROMs for rom hacking purpo… X.C.O.P.Y. → **Until Now :P**

- Released by 'yumeji' in 2007, but website n… available (geocities.jp).
- Need to dig on shady forums to find a working copy



Support CPS2 crypto in radare2

- Take the CPS2 decryption algorithm from MAME
  - MAME: src/mame/machine/cps2crypt.cpp
- Add it to rahash2
  - r2: libr/crypto/p/crypto_cps2.c
- Invert the feistel to also support encryption

```
// de/en-crypt the opcodes
for (a = i; a < length/2 && a < upper_limit/2; a += 0x10000) {
        if (crypt_direction) {
                /* decrypt */
                dec[a] = feistel (rom[a], fn2_groupA, fn2_groupB,
                        &sboxes2[0*4], &sboxes2[1*4], &sboxes2[2*4], &sboxes2[3*4],
                        key2[0], key2[1], key2[2], key2[3]);
                dec[a] = r_read_be16 (&dec[a]);
        } else {
                /* encrypt */
                dec[a] = r_read_be16 (&rom[a]);
                dec[a] = feistel (dec[a], fn2_groupA, fn2_groupB,
                        &sboxes2[3*4], &sboxes2[2*4], &sboxes2[1*4], &sboxes2[0*4],
                        key2[3], key2[2], key2[1], key2[0]);
        }
}
```

- Finally write test cases for *radare2-regressions* ;)

Decrypt, patch, encrypt a ROM (1)

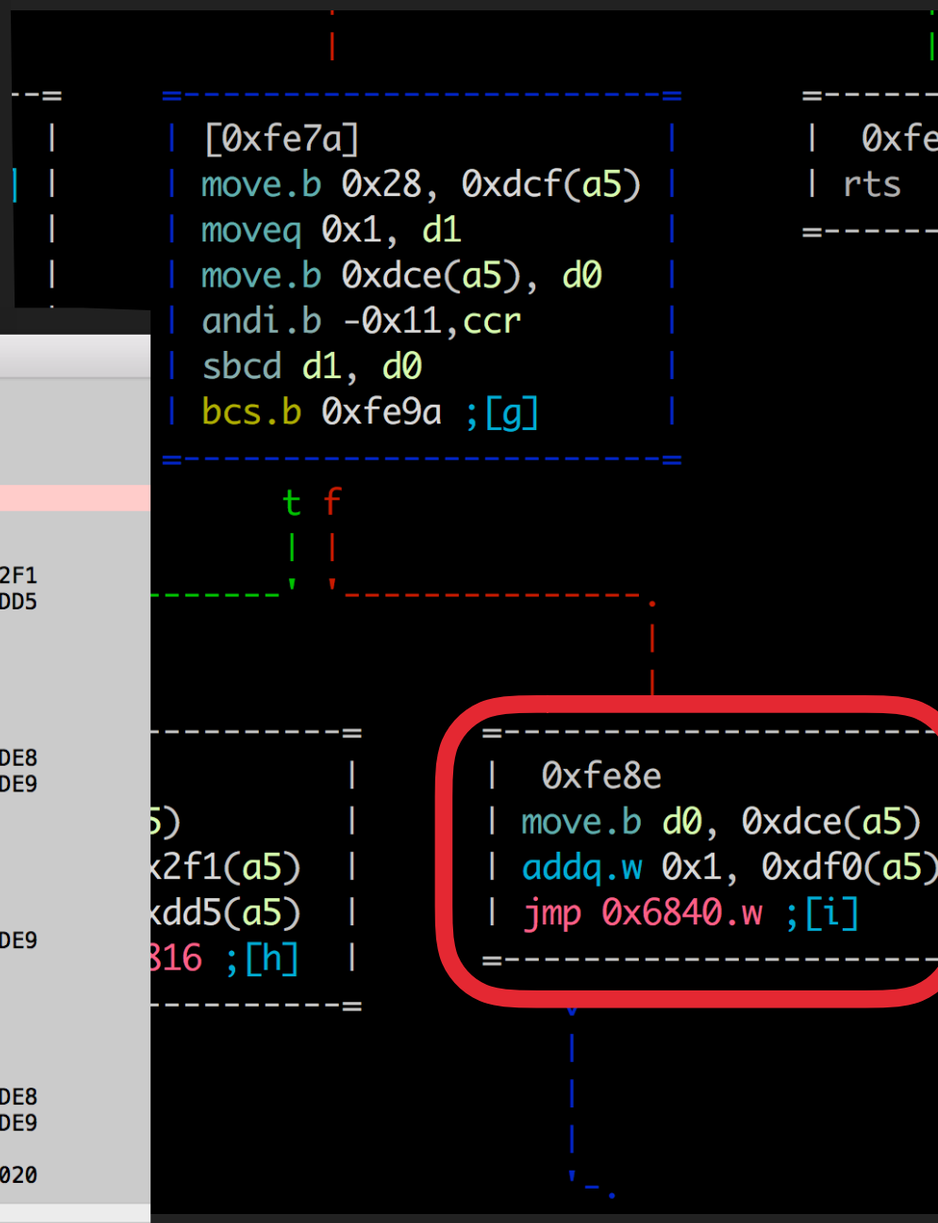
MAME: Super Street Fighter II X: Grand Master Challenge (Japan 940223) [ssf2xj]

Debug: ssf2xj - M68000 ':maincpu'

cycles	376
beamx	257
beamy	3
frame	3658
flags	.....
PC	00FE92
SP	00FF0484
ISP	FFFFFFF6
USP	00FF0484
D0	00000068
D1	00000001
D2	0000001E
D3	0000FFFF
D4	00000030
D5	00000000
D6	000001AB
D7	00000000
A0	00708D60
A1	FFFF82C0
A2	0002F586
A3	0010229C
A4	FFFF9362
A5	FFFF8000
A6	FFFF0080
A7	00FF0484
PREF_ADDR	00FE92
PREF_DATA	0000526D

```
00FE86  andi    #$ef, CCR          023C 00EF
00FE8A  sbcd    D1, D0             8101
00FE8C  bcs     $fe9a              650C
00FE8E  move.b  D0, ($dce,A5)      1B40 0DCE
00FE92  addq.w  #1, ($df0,A5)      526D 0DF0
00FE96  jmp     $6840.w            4EF8 6840
00FE9A  move.b  ($dce,A5), D0      426D 0DCE
00FE9E  move.b  #$1, ($2f1,A5)     1B7C 0001 02F1
00FEA4  move.b  #$1, ($dd5,A5)     1B7C 0001 0DD5
00FEAA  jmp     $6816.w            4EF8 6816
00FEAE  move.b  ($de8,A5), D0      102D 0DE8
00FEB2  move.w  ($6,PC,D0.w), D1   323B 0006
00FEB6  jmp     ($2,PC,D1.w)       4EFB 1002
00FEBA  ori.b   #$18, D6           0006 0018
00FEBE  ori.b   #$2d, INVALID 32   0032 542D 0DE8
00FEC4  move.b  #$4, ($de9,A5)     1B7C 0004 0DE9
00FECA  move.w  #$d, D3            363C 000D
00FECE  bra     $ff02              6000 0032
00FED2  subq.b  #1, ($de9,A5)      532D 0DE9
00FED6  bne     $feea              6612
00FED8  addq.b  #2, ($de8,A5)      542D 0DE8
00FEDC  move.b  #$2, ($de9,A5)     1B7C 0002 0DE9
00FEE2  move.w  #$e, D3            363C 000E
00FEE6  bra     $ff02              6000 001A
00FEEA  rts                        4E75
00FEEC  subq.b  #1, ($de9,A5)      532D 0DE9
00FEF0  bne     $feea              66F8
00FEF2  move.b  #$2, ($de8,A5)     1B7C 0002 0DE8
00FEF8  move.b  #$4, ($de9,A5)     1B7C 0004 0DE9
00FEFE  move.w  #$d, D3            363C 000D
00FF02  lea     $709020.l, A0      41F9 0070 9020
00FF08  move.w  D3, ($6,A0)        3143 0006
```

```
>bpset 0xfe8e
Breakpoint 1 set
>go
Stopped at breakpoint 1
>go
Stopped at breakpoint 1
>go
Stopped at breakpoint 1
>go
```



Decrypt, patch, encrypt a ROM (2)

```
[0x00000000]> b 25
[0x00000000]> pD @0xfe8e
        |   0x0000fe8e      1b400dce        move.b d0, 0xdce(a5)
        |   0x0000fe92      526d0df0        addq.w 0x1, 0xdf0(a5)
        └─< 0x0000fe96      4ef86840        jmp 0x6840.w                ; jump
            0x0000fe9a      426d0dce        clr.w 0xdce(a5)
            0x0000fe9e      1b7c000102f1    move.b 0x1, 0x2f1(a5)
            0x0000fea4      1b7c00          move.b 0, 0(a5)
[0x00000000]> wx 4e714e71@0xfe8e
[0x00000000]> pD @0xfe8e
        |   0x0000fe8e      4e71            nop                         ; no operation
        |   0x0000fe90      4e71            nop                         ; no operation
        |   0x0000fe92      526d0df0        addq.w 0x1, 0xdf0(a5)
        └─< 0x0000fe96      4ef86840        jmp 0x6840.w                ; jump
            0x0000fe9a      426d0dce        clr.w 0xdce(a5)
            0x0000fe9e      1b7c000102f1    move.b 0x1, 0x2f1(a5)
            0x0000fea4      1b7c00          move.b 0, 0(a5)
[0x00000000]>
```

- `$ rahash2 -D cps2 -S "0x942a5702 0x05ac140e" sfxj.03c > d_sfxj.03c`
- `$ r2 -qwn -c "wx 4e714e71@0xfe8e" d_sfxj.03c # infinite time`
- `$ rahash2 -E cps2 -S "0x942a5702 0x05ac140e" d_sfxj.03c > sfxj.03c`

DEMOS

- DEMO 1
  - Infinite time: `wx 4e714e71 @ 0xfe8e`

- DEMO 2
  - Jedpossum Training Mode:

- `$ rahash2 -D cps2 -S "0x942a5702 0x05ac140e" sfxj.03c > d_sfxj.03c`
- `$ rahash2 -D cps2 -S "0x942a5702 0x05ac140e" sfxj.04a > d_sfxj.04a`
- `$ r2 -qwn d_sfxj.03c < patch_03c.txt`
- `$ r2 -qwn d_sfxj.04a < patch_04a.txt`
- `$ rahash2 -E cps2 -S "0x942a5702 0x05ac140e" d_sfxj.03c > sfxj.03c`
- `$ rahash2 -E cps2 -S "0x942a5702 0x05ac140e" d_sfxj.04a > sfxj.04a`

Future work

- Fix hardcoded UPPER_LIMIT value: currently set to 0x400000
- Support CPS3 encryption: I really haven't looked into it yet

Questions?

THANK YOU!

Bibliography

- http://en.wikipedia.org/wiki/CP_System_II
- http://cps2shock.emu-france.info/
- http://forums.shoryuken.com/discussion/169077/hacking-the-st-rom/p1
- http://www.mamecheat.co.uk/forums/viewtopic.php?p=13271#p13271
- http://andreasnaive.blogspot.com.es/2006_12_01_archive.html
- http://andreasnaive.blogspot.com.es/2007_01_01_archive.html
- http://pof.eslack.org/2014/04/22/ssf2t-the-quest-for-the-perfect-training-mode/